Robot Motion Planning among Moving Obstacles

1'solo Fiorini* arid ZviShiller†

* Jet Propulsion Laboratory
California Institute of Technology
Pasadena. CA 91109 USA

† Department of Mechanical. Nuclear and Aerospace Engineering
University of California. Los A ngeles
Los Angeles. CA 90024 USA

Al)str-net

This paper presents an on-line method for computing the motion of a robot in a dynamic environment subject to the robot dynamics and its actuator constraints. This method is based 011 the concept of Velocity Obstacle that defines the set of robot velocities that would result in a collision between the rob ot and an obstacle moving at a given velocity. The avoidance maneuver at a specific time is thus computed by selecting robot's velocities out of that set. The set of all avoiding velocities is reduced to the dynamically feasible maneuvers by considering the robot "s acceleration constraints. This computation is repeated at regular time intervals to account for general obstacle trajectories.

The trajectory consists of a sequence of avoidance maneuvers that are computed at discrete time intervals using the concept of Velocity Obstacle. i.e. selecting those velocit ics that avoid future collisions and satisfy the dynamic constraints. The trajectory from start to goal is then composed by searching the tree of the avoi dance maneuvers. A n exhaustive search of the tree yields near-optimal trajectories that either minimize distance or motion time. For on-time applications, a heuristic scare]] of the tree that selects one avoidance maneuver at each time int erval is proposed to satisfy a pri oritized list Of objectives, such as reaching the goal or maximizing speed.

A dynamic optimization is used to verify the quality of the trajectory computed by the heuristic search. It uses the trajectory as its initial guess, eliminates the effects of the time discretizati on. and minimizes motion time. subject to the dynamics of the robot. its actuator limits and the state inequality constraints due to the moving obstacles. This step establishes a criterion for evaluating the performance of ol)-line planning methods for dynamic environments.

This approach is demonstrated for planning the trajectory of an automated vehicle in an Intelligent Vehicle Highway System scenario.

1. Introduction

This paper addresses the problem of motion planning in dynamic environments. Typical examples of dynamic en vironm e nts include manufacturing tasks in whi ch robot manipulators track and retrieve parts from moving conveyers. and air. sea. and land traffic. where aircraft. vessels and vehicles avoid each other while moving towards their dcst.ills.i.ion.

Motion planning in dynamic environments is considerably more difficult than the widely studied static problem. since it requires the simultaneous solution of the path planning and of the velocity planning problems. Path planning involves the computation of a collision free path from start to goal without considering robot dynamics. Velocity planning. on the ot her hand. involves the computation of the velocity profile along a given path. satisfying system dynamics and actuator constraints. In addition. motion planning in static environments can be guaranteed to find a solution if one exists at time t_0. whereas motion planning in dynamic environments is essentially intractable. [36]. [9]. i.e. the solution at t_0 may not exist at a later time because of the evolution of the environment.

Previous methods consisted of a graph search in a position-time configuration space of the robot [12.] 3], and checking for intersection of the swept volumes of the robot and obstacles in a Cartesian-time

space [5- 8]. Velocity constraints were considered in [35] for the solution of the asteroid problem, and acceleration constrains were satisfied in [1 0] and in [33] for planning for a point mass.

Approximate dynamic constraints were satisfied in [23.24] by decomposing the planning problem into path and velocity planning, and by limiting slope, direction and curvature of the trajectory in a position-time plane. Dynamic constraints were used in [1 9,20] to define the collision front, and in [1 7,18] to introduce the concept of transient obstacles, also applicable to dynamic environments.

To date, on-line dynamic planning has been treated by emphasizing reasoning and decision making [36], or by creating artificial potential fields around the obstacles [2 5].

The time-optimal motion planning problem in static environments has been treated previously using parameter optimizations, representing the trajectory as a polynomial in time and accounting for obstacles using a penalty function [22]. and representing the path by a cubic spline [38]. [1]. computing the motion time along the path with an efficient method developed in [1] and in [39]. Time optimal motion planning of cooperating robots moving along specified paths has been studied in [27]. [39].

In this paper, we develop an efficient method for computing the trajectories of a robot moving in a time-varying environment. It utilizes the concept of Velocity Obstacle (VO), which represents the robot's velocities that would cause a collision with an obstacle at some future time. An avoidance maneuver is computed by selecting velocities that are outside of the velocity obstacle. To ensure that the maneuver is dynamically feasible, robot dynamics and actuator constraints are mapped into the robot velocity space. A trajectory consists of a sequence of such avoidance maneuvers, computed by searching over a tree of avoidance maneuvers generated at discrete time intervals. For on-line applications, the tree is pruned using a heuristic search designed to achieve a prioritized set of objectives, such as avoiding collisions, reaching the goal, maximizing speed, or computing trajectories with desirable topology.

The solutions computed with this method are conservative, since they exclude trajectories that, although feasible, include avoidance maneuvers violating the velocity obstacle. To evaluate the quality of these trajectories, they are compared to the trajectories computed using a dynamic optimization, which are not bound by the velocity obstacle approach. The optimization used here is a steepest descent [3]. [1 1]. modified to include the state-dependent inequality constraints due to the moving obstacles. The initial guess of the optimization is the trajectory generated by the search over the maneuver tree.

The advantages of this approach are multi-fold: *i*) it permits an efficient geometric representation of potential avoidance maneuvers of the moving obstacles, *ii*) any number of moving obstacles can be avoided by considering the union of their VO's, *iii*) it unifies the avoidance of moving as well as stationary obstacles, and *iv*) it allows for the simple consideration of robot dynamics and actuator constraints.

This paper is organized as follows. Section 2 defines the Velocity Obstacle, the avoidance velocities, and describes the representation used to compute the trajectories. Section 3 describes the dynamic optimization used to compute the time optimal trajectories. Examples of trajectories, and of their corresponding time optimal solutions are presented in Section 4.

2. The Velocity Obstacle

The Velocity **Obstacle** (VO) is an extension of the Configuration Space Obstacle [30] to a time-varying environment. It consists of the velocities of the robot that will cause a collision between the robot and the obstacles at some future time. Although this concept is valid for general robots and obstacles, in this paper we restrict our analysis to circular robots and obstacles in the plane.

In this section we first define the VO concept and then we combine it with the dynamic constraints of the robot to compute avoidance velocities that are also dynamically feasible. The corresponding avoidance maneuvers are used to build a tree that represents, within the given temporal resolution, all the avoidance trajectories generated to satisfy the constraints represented by the velocity obstacles.

2.1. Definition of Velocity Obstacle

The VO is illustrated using the scenario shown in Figure 1, where two circular objects, A and B_1, are shown at time t_0 with velocities $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$. Circle A represents the robot, and circle B_1 represents the obstacle. The velocities and positions of A and B_1 were chosen so that A and B_1 will collide at some time t_1, $(t_1 > t_0)$, provided that $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$ do not change.

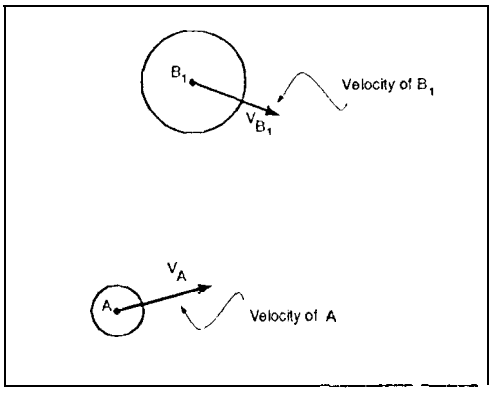

Figure 1: The robot and a moving obstacle.

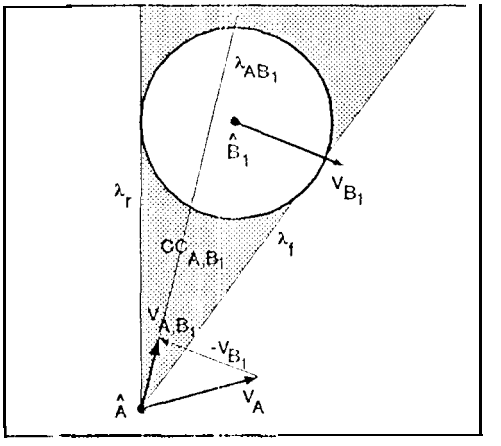

Figure 2: The Collision Cone CC_{A,B_1}.

To compute the VO, we first represent B_1 in the *Configuration Space* of A by reducing A to the point $\widehat{A}$, and enlarging B_1 by the radius of A to the circle $\widehat{B_1}$ [29], and then we attach the velocity vectors to the position of $\widehat{A}$ and to the center of $\widehat{B_1}$, respectively.

By considering the relative velocity $\mathbf{v}_{A,B_1} = \mathbf{v}_A - \mathbf{v}_{B_1}$, and by assuming that A and B_1 maintain their current velocities, a collision between $\widehat{A}$ and $\widehat{B_1}$ will occur at some future time $t_1 > t_0$ if the line λ_{A,B_1} of the relative velocity $\mathbf{v}_{A,B_1}$ intersects $\widehat{B_1}$. In fact, any relative velocity that lies between the two tangents to $\widehat{B_1}$ λ_f and λ_r will cause a collision between A and B_1. Therefore, we define the *Collision Cone*, CC_{A,B_1}, as the set of colliding relative velocities between $\widehat{A}$ and $\widehat{B_1}$ or:

$$CC_{A,B_1} = \{\mathbf{v}_{A,B_1} \mid \lambda_{A,B_1} \cap \widehat{B_1} \neq \emptyset\} \tag{1}$$

This cone is the planar sector with apex in $\widehat{A}$, bounded by the two tangents λ_f and λ_r from $\widehat{A}$ to $\widehat{B_1}$, as shown in Figure 2.

The collision cone thus partitions the space of *relative* velocities into colliding and avoiding velocities. The relative velocities, $\mathbf{v}_{A,B_1}$, lying on the boundaries of CC_{A,B_1} represent *tangent* maneuvers that would graze the obstacle B_1.

The collision cone is specific to a particular pair of robot/obstacle. To consider multiple obstacles, it is useful to establish an equivalent partition of the *absolute* velocities of A. This is done simply by adding the velocity of B_1, $\mathbf{v}_{B_1}$, to each velocity in CC_{A,B_1} or, equivalently, by translating the collision cone CC_{A,B_1} by $\mathbf{v}_{B_1}$, as shown in Figure 3 [15]. The *Velocity Obstacle* VO is then defined as:

$$VO = CC_{A,B_1} \oplus \mathbf{v}_{B_1} \tag{2}$$

where $\oplus$ is the Minkowski vector sum operator.

Thus, the VO partitions the absolute velocities of A into *avoiding* and *colliding* velocities. Velocities on the boundaries of VO would result in A grazing B_1, since the corresponding relative velocities lie on the boundary of the collision cone CC_{A,B_1}. Note that the VO of a stationary obstacle is identical to its relative velocity cone, since then $\mathbf{v}_{B_1} = 0$.

To avoid multiple obstacles, the VO's of each obstacle are combined into a single velocity obstacle:

$$VO = \cup_{i=1}^{m} VO_i \tag{3}$$

where m is the number of obstacles. The VO assumes that the velocity of B_1 remains constant. To account for variable velocities, VO is recomputed at specified time intervals.

The assumption of circular robot and obstacles reduces the dimension of the configuration space, and thus greatly simplifies the computation of the VO. It also fixes the shape of the configuration space obstacles, which are generally functions of their positions in the robot's work space. For general manipulators, the VO must be periodically recomputed to account for the time varying configuration space obstacles [13].

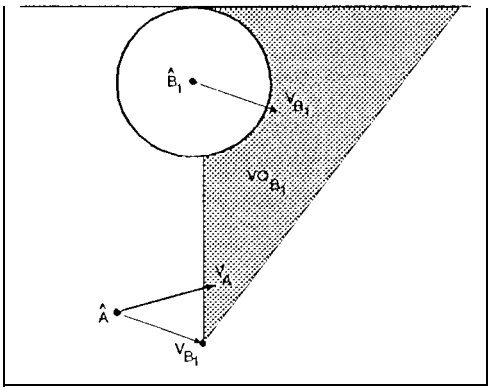

Figure 3: The velocity obstacle VO.

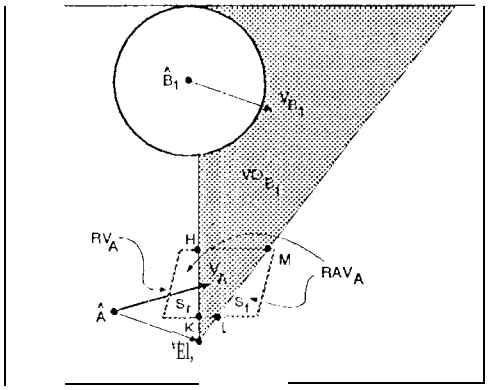

Figure 4: The reachable avoidance velocities.

2.2. The Avoidance Maneuvers

The velocities reachable by robot A at a given state over a given time interval Δt are computed by transforming the dynamic constraints of the robot into bounds on its acceleration. The set of *feasible accelerations* at time t_0, $FA(t_0)$, is *defined* as;

$$FA(t_0) = \{\mathbf{x} \mid \ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}), \mathbf{u} \in U\} \tag{4}$$

where $f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ represents the dynamics of the robot. $\mathbf{u}$ are the actuator efforts, U is the set of admissible controls, and x is the position vector defined earlier. Note that the feasible acceleration range of a two degree-of-freedom system with decoupled actuator limits is a parallelogram **[37].**

The set of *reachable* velocities, $\mathrm{RV}(t_0 + \Delta t)$, over the time interval Δt is thus defined as:

$$\mathrm{RV}(t + \Delta t) = \{\mathbf{v} \mid \mathbf{v} = \mathbf{v}_A(t_0) \oplus \Delta t \cdot FA(t_0)\} \tag{5}$$

The set of *reachable avoidance* velocities, RAV, is defined as the difference between the reachable velocities and the velocity obstacle:

$$\mathrm{RAV}(t_0 + \Delta t) = \mathrm{RV}(t_0 + \Delta t) \ominus VO(t_0) \tag{6}$$

where $\ominus$ denotes the operation of set difference. A maneuver avoiding obstacle B_1 is thus computed by selecting any velocity in RAV. Figure 4 shows schematically the reachable velocity set RAV consisting of two disjoint closed sets, S_f and S_s. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

2.3. Computing the avoidance trajectories

The trajectory that avoids static and moving obstacles, reaches the goal, and satisfies the robot's dynamic constraints is computed as a discrete sequence of elementary avoidance maneuvers, selected by a global search over the tree of all feasible maneuvers at specified time intervals. Alternatively, the global search may be reduced to a heuristic search for on-line applications, where the trajectories of the moving obstacles are hot know a-priori, but are rather acquired in real-time. These two approaches are discussed next.

2.3.1. Global Search

We represent the state space of the robot by a tree of avoidance maneuvers at discrete time intervals. The *nodes* on this tree correspond to the positions of the robot at discrete times t_i. The *operators* expanding a node at time t_i into its successors at time $t_{i+1} = t_i + T$ are the velocities in the reachable

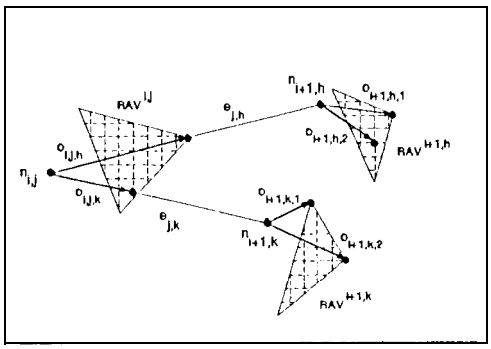

Figure 5: Representation for global search.

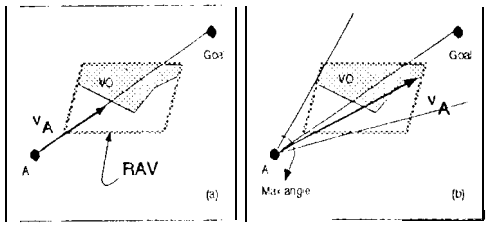

Figure 6: a: TG strategy. b: MV strategy.

avoidance velocity set RAV. The *edges* correspond to the avoidance maneuvers at those positions [34,32]. The search tree is then defined as follows:

$$n_{i,j} = \{\mathbf{x}_{i,j} = (x_j(t_i), y_j(t_i)), \mathbf{v}_{i,j} = (v_{j,x}(t_i), v_{j,y}(t_i))\} \quad (7)$$

$$o_{i\ j,l} = \{\mathbf{v}_l \mid \mathbf{v}_l \in \mathrm{RAV}_j(t_i)\} \quad (8)$$

$$e_{j,\ k} = \{(n_{i,j}, n_{i+1,k}) \mid n_{i+1,k} = n_{i,j} + (o_{i,j,l}\,\Delta t)\} \quad (9)$$

where $n_{i,j}$ if the jth node at time t_i. $\mathrm{RAV}_j(t_i)$ is the reachable velocity set computed for node $n_{i,j}$. $o_{i,j,l}$ is the lth operator on node j at time t_i, and $e_{j,k}$ is the edge between node n_j at time t_i, and node n_l at time t_{i+1}.

The tree of all feasible avoidance maneuvers is constructed as follows. At time t_i, the avoidance set RAV, corresponding to node $n_{i,j}$, is discretized by a grid. Then, the velocities corresponding to each node on the grid are used to compute the edges emanating from node n_iThe positions reached by the robot at the end Of each maneuver are the successors of node $n_{i,j}$. A node $n_{i,j}$ is completely expanded when all the operators $o_{i,j,l}$ have been applied and all the edges emanating from $n_{i,j}$ have been examined. The resulting tree has a constant time interval between nodes, a variable branch number that is a function of the shape of each RAV, and it can be searched using standard techniques [34,32]. Figure 5 snows schematically a subtree of some avoidance maneuvers.

Since the avoidance maneuvers are selected based on the velocity obstacles VO, they are never on a collision course with any of the considered obstacles, as discussed earlier. This excludes trajectories Ills.1 might, part of the time, be on a collision course with some of the obstacles. However, such trajectories may be generated by either considering only obstacles with imminent collisions, or by refining the trajectory using a dynamic optimization, as discussed later in Section 3.

2.3.2. Heuristic Search

For **ol)-line** applications, or when only incomplete information about the environment is available, the maneuver tree can be constructed incrementally using heuristic rules designed to satisfy the prioritized set of gods embedded in the formulation of the velocity obstacle. The survival Of the robot can be guaranteed by selecting the avoidance velocities RAV; the target can be reached by selecting velocities I.list point towards the destination; the motion time can be minimized by choosing the highest velocity available; and the desired trajectory structure can be selected by choosing an appropriate sequence of front and rear avoidance maneuvers [14].

We thus propose the following basic heuristics: (a) TG (1.0 goal), choose the highest avoidance velocity along the line to the goal, as shown in Figure 6-a; (b) MV (maximum velocity), select the maximum avoidance velocity within some specified angle α from the line to the goal, as shown in Figure 6-b. Other heuristics may combine the above strategies in order to better satisfy the prioritized goals.

3. The Dynamic Optimization

The trajectory computed by the heuristic search is evaluated off-line by a dynamic optimization. This optimization is formulated using Mayer notation for the performance index J:

$$\min_{u(t)\in \mathbf{U}} J \quad , \quad \text{with } J = t_f = \phi(\mathbf{x}(t_f), t_f) \tag{10}$$

subject to the following kinematic and dynamic constraints:

$$(1)\ \Gamma(\mathbf{x}(t_0), t_0) = 0 \qquad (2)\ \Omega(\mathbf{x}(t_f), t_f) = 0 \qquad (3)\ \Psi : \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) = 0] \tag{11}$$

$$(4)\ \dot{x} = \text{Y-(X.11)} = \text{f(x) -t } \mathbf{g(x)u} \qquad (5)\ \mathbf{U} = \{\mathbf{u} \mid u_i(min) \le u_i \le u_i(max)\} \tag{12}$$

where (1) is the initial manifold, (2) is the terminal manifold, (3) represents the time-varying obstacles, (4) is the robot dynamics, and (5) represents the admissible controls.

State constraints due to the presence Of obstacles (11 -3) are differentiated with respect to time until they become explicit in the controls $\mathbf{u}$, and then appended as state dependent control constraint to the Hamiltonian [4]. [11]. The number of differentiations of each constraint represents the *order* p of that constraint. This approach requires an additional tangency constraint at the entry point of the constrained arc [40]. This may over-cmlsirain the problem for constraints of order higher than two *[21].*

For' example, in the case of a single obstacle, the state constraint (113) is replaced by the tangency condition, denoted Ψ_1, and the control equality constraint, denoted Ψ_2:

$$\Psi_1 : \begin{pmatrix} S(\mathbf{x}(t), t) = 0 \\ \dot{S}(\mathbf{x}(t), t) = 0 \\ \vdots \\ S^{(p-1)}(\mathbf{x}(t), t) = 0 \end{pmatrix} \quad t = t_{1} \quad , \quad \Psi_2 : \ S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \qquad t_1 \le t \le t_2 \tag{13}$$

Then, the admissible control set $\mathcal{U}$ for the optimal control $u^*(t)$ becomes:

$$\mathcal{U} : \begin{cases} \mathbf{u} : \mathbf{u}_{min}(\mathbf{x}) \le \mathbf{u} \le \mathbf{u}_{max}(\mathbf{x}) \\ S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \ \text{ for } S(\mathbf{x}, t) = 0 \end{cases} \tag{14}$$

3.1. The Necessary Optimality Conditions

The optimal control $\mathbf{u}^*(t)$ in the interval $t_0 \le t \le t_f$, that generates the optimal solution, $\mathbf{x}^*(t)$, minimizes J, and satisfies the fixed terminal manifolds Γ and Ω, is computed by satisfying the necessary conditions of Pontryagin Minimum Principle [4], [2], [28].

The Hamiltonian function $\mathcal{H}$ for this Problem is defined as [4] [28]:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T(\mathbf{f(x)} + \mathbf{g(x)u}) \text{ -t } \mu^T \varphi(\mathbf{x}, \mathbf{u}) \tag{15}$$

where Λ and μ are vectors of Lagrange multipliers, and $\varphi(\mathbf{x}, u)$ is the set of active control constraints at time t, $t_0 \le t \le t_f$.

The adjoint equations for the Lagrange multipliers are given by:

$$\dot{\Lambda} = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T \qquad \Lambda^T(t_f) = \left(\frac{\partial \phi}{\partial x} + \nu^T \frac{\partial \Omega}{\partial x}\right)_{t_f} \tag{16}$$

which exhibit the discontinuity at the entry point of the constraint given by [11]:

$$\Lambda^T_{t_1^-} = \Lambda^T_{t_1^+} + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)} \tag{17}$$

When the constraint (13) is active, the minimization of the Hamiltonian for an autonomous system is equivalent to satisfying:

$$\mathcal{H}^*_u(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \ t_0 \le t \le t_f \tag{18}$$

and the adjoint equations for the constrained arcs become:

$$\dot{\Lambda}^T = -\Lambda^T \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}(\mathbf{x})\mathbf{u}}{\partial \mathbf{x}} - \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \varphi(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} \right] \tag{19}$$

in summary, the trajectory minimizing the performance index $J = t_f$ is characterized by the following: (i) its controls $\mathbf{u}^*$ are in the admissible set U. *(ii)* the states satisfy the terminal manifolds Γ at t_0 and Ω at t_f, *(iii)* the co-state equations are described by (16) on the free arcs and by (19) on the constrained arcs, with a discontinuity at the junction of the two given by (17), and *(iv)* the Hamiltonian is minimized over the entire interval. These conditions are satisfied by the trajectory computed by the numerical method described next [14].

3.2. Numerical Computation

The optimal trajectory is computed numerically by appending the constraints to the performance index J via appropriate arrays of Lagrange multipliers, and by computing the corrections to the controls that drive to zero the differential of the augmented performance index $d\hat{J}$ [14]. This differential is then derived as function of the variations in the control switches, assuming a bang-bang solution [41]. Singular arcs on the solution are then approximated by a finite number of switches generating a quasi-optimal solution [31].

The differential of the augmented performance index is formed by computing the differentials of J, Ω and Ψ_1 independently, then including the effects of Ψ_1 on dJ and $d\Omega$, and finally combining all into $d\hat{J}$. The final form of the augmented performance index is:

$$d\hat{J} = \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f + \int_{t_0}^{t_1^-} \mathcal{H}_u \delta u d\tau + \int_{t_1^+}^{t_f} \mathcal{H}_u \delta u d\tau \tag{20}$$

where the Hamiltonian is given by $\mathcal{H} = \Lambda^T \mathcal{F} + \mu\varphi$ as in equation (15); the Lagrange multiplier Λ is defined as $\Lambda^T = \lambda_\phi^T + \nu^T \lambda_\Omega^T + \eta^T \lambda_\Psi^T$, since the constraints Ψ_1 are not affected by the state after t_1; and η, and ν are two constant Lagrange multipliers.

Bang-bang controls can only assume one of two values α_M or α_m, and the variation in $\mathbf{u}$ is rewritten as $\delta u_i = (\alpha_M - \alpha_m)\, sgn(dt_i)$, where sgn is the signum function.

If $s_{s,r}$ indicates the number of switches in the s segment of the trajectory for the rth input, and m indicates the number of elements in $\mathbf{u}$, then $d\hat{J}$ for bang-bang controls becomes:

$$\begin{aligned} d\hat{J} = & \sum_{i=1}^{m} \sum_{j=1}^{s_{1,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} \\ & + \sum_{i=1}^{m} \sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} + \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f \end{aligned} \tag{21}$$

This differential is minimized by choosing the steepest descent increments as [4]:

$$dt_{ij} = -\frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii} \delta u_i} \qquad dt_f = -\frac{1}{b} \left(\mathcal{H} + \frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} \right)_{t_f} \tag{22}$$

with w_{ii} and b being suitable positive values.

The values of dt_{ij} and dt_f depend on multipliers η and ν. Their values are computed by requiring that differentials $d\Psi(t_1)$ and $d\Omega(t_f)$ be reduced by given quantities c and θ. The values for the multipliers ν and η are:

$$\eta = -I_{\Psi\Psi}^{-1} (c + I_{\Psi\Omega}\nu + I_{\Psi\phi}) \tag{23}$$

$$\nu = -\left({}^1I_{\Omega\Omega} + {}^1I_{\Omega\Psi}\, {}^1I_{\Psi\Psi}^{-1}\, {}^1I_{\Psi\Omega} + {}^2I_{\Omega\Omega} + {}^3I_{\Omega\Omega} + \frac{1}{b} \left(\frac{d\Omega}{dt} \frac{d\Omega}{dt}^T \right)_{t_f} \right)^{-1}$$

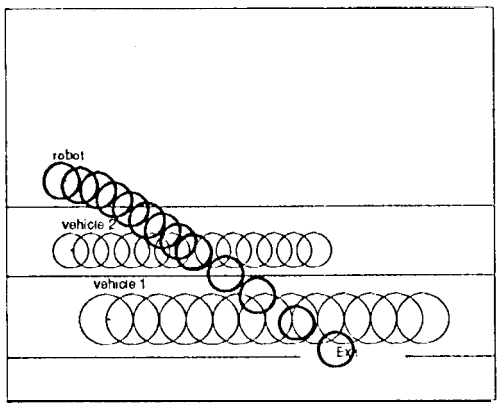

Figure 9: Solution with the TG strategy

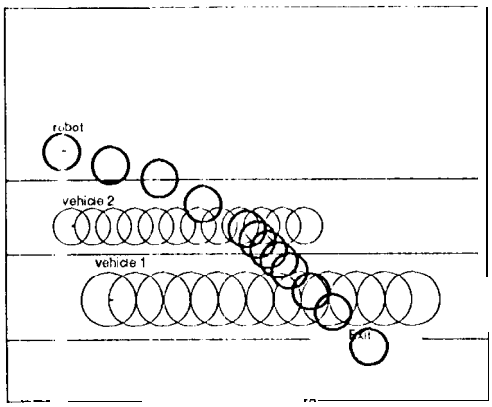

Figure 10: Solution with TG-MV strategies

4.1. The VO Trajectory

The trajectory shown in Figure 8 is computed using the Depth First Iterative Deepening algorithm [26]. This algorithm returns the fastest path to the target, the first time it reaches the target. In the example, the RAV sets have been discretized by considering four points on each side of the boundary, and the maximum feasible velocity in the direction to the goal. The search has been expanded to 4.5 s and a depth of 5 levels. The total motion time of this solution is of 3.5 s.

Using the TG strategy resulted in the trajectory shown in Figure 9. Along this trajectory, the robot slows down and lets vehicle 2 pass, and then speeds up towards the exit, behind vehicle 1. The total motion time for this trajectory is 6.07 s.

The trajectory computed using both MV and TG strategies is shown in Figure 10. Along this trajectory, the robot first speeds up to pass vehicle 2, then slows down to let vehicle 1 pass on, and then speeds up again towards the goal. The motion time for this trajectory is 5.31 s. This trajectory was computed by using the MV heuristics for $0 \leq t \leq 2.0$ s and the TG heuristics afterwards.

4.2. The Optimal Trajectory

Figure 11 shows the initial guess of the optimization, obtained by integrating the bang-bang controls. The effects of approximating the trajectory of Figure 10 show in the different terminal position of the robot, far from the target, and in the collision with vehicle 1. This collision is represented in Figure 11 by the black marks between time $t = 2.5$ s and time $t = 3.0s$. The marks represent the depth of the penetration of the robot into vehicle 1.

This trajectory resembles the trajectory of Figure 10, namely the type of maneuvers used in avoiding the obstacles, and the time of the avoidance of vehicle two, at approximately $t = 2.5s$. Then, using this trajectory as the initial guess of the dynamic optimization, the optimal solution satisfies the kinematic and dynamic constraints of the problem.

The optimal solution is shown in Figure 12, and it has the same key features of the heuristic trajectory of Figure 10, thus allowing for a meaningful comparison between the trajectory computed by the heuristic search and the corresponding optimal solution. The motion time of the optimal trajectory is 4.6 s, which compares favorably with the motion time of the heuristic solution of 5.31 s.

5. Conclusion

A novel method for planning the motion of a robot moving in a time-varying environment has been presented. It is significantly different from currently available planning algorithms, since it simultaneously computes the path and velocity profile. It can avoid all static and moving obstacles and satisfy the robot's dynamic constraints.

The method consists of computing, for every obstacle, its corresponding velocity obstacle, which is the set of colliding velocities between the obstacle and the robot. Then, by subtracting it from the reachable velocities of the robot, the set of reachable avoidance velocities is formed, which consists of all

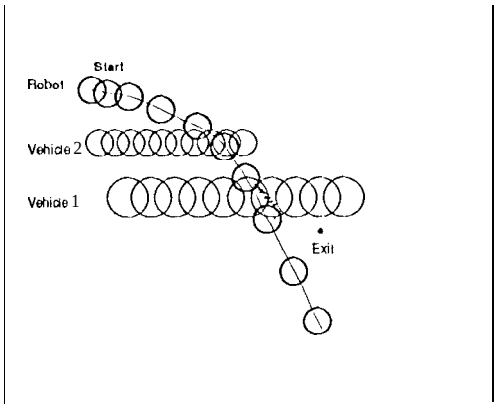

Figure 11: Initial guess of the optimization.

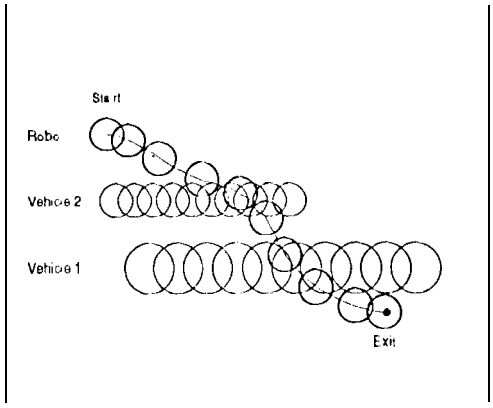

Figure 12: Optimal solution.

the velocities that avoid the obstacles and satisfy the robot's dynamic constraints. A search space is then formed by representing the state space of the robot by a tree of avoidance maneuvers. A global search over the tree yields trajectories that minimize a selected performance index, such as motion time or traveled distance. The solutions computed with this method are conservative, since each maneuver avoids all obstacles, irrespectively of their expected collision time, and they may exclude trajectories that, although feasible, violate the velocity obstacle in some interval. For on-line applications, the tree is pruned using one of several heuristic strategies, aimed at satisfying a prioritized list of goals, such as the survival of the robot, reaching the target, and minimize motion time. The quality of the heuristic trajectories is evaluated by comparing them to the optimal trajectories computed by dynamic optimization. The method is demonstrated for planning the trajectory of an automated vehicle in an Intelligent Vehicle Highway System scenario.

The main advantages of the velocity obstacle approach include the efficient geometric representation of maneuvers avoiding any number of moving and static obstacles, and the simple consideration of robot dynamics and actuator constraints. This approach, therefore, makes it possible to Compute on-line safe and feasible trajectories for robots in dynamic environments.

6. Acknowledgment

The research described in this paper has been partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautic and Space Administration.

REFERENCES

1. J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3 17, Fall 1985.
2. A.E. Bryson and S.E. Denham, W.F. and Dreyfus. Optimal programming problems with inequality constraints *i*: Necessary conditions for extremal solutions. *AIAA Journal*, 1(11):2544 2550, November 1963.
3. All. Bryson and W.F. Denham. A steepest-ascent, method for solving optimum programming problems. *ASME Journal of Applied Mechanics*,(29):247 *257.* June 1962.
4. Al?. Bryson and Y.C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York. NY, *1975.*
5. *S.* Cameron. A study of the clash detection problem in robotics. In *IEEE International Conference on Robotics and Automation*, pages 488 *493,* St. Louis. MO, March *25-281985.*
6. *S.* Cameron. Efficient intersection tests for objects defined constructively. *The International Journal of Robotics Research*, 8(1):3 25, February 1989.
7. *S.* Cameron. Collision detection by four-dimensional intersection testing. *IEEE Journal of Robotics and Automation*, 6(3):291 302, June 1990.
8. S.A. Cameron. *Modelling Solids in Motion.* PhD thesis, Edinburgh, UK. 1984.
9. J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *28th IEEE Symposium on Foundation of* ~O?l/Jill.t *er Science*, 1987.

10. J. Canny, J. Reif, B. Donald, and P. Xavier. On the complexity of kinodynamic planning. In *29th IEEEE Symposium on Foun dation of Comput er Scien ce*, 1988.
11. A. I?. Denham, W .F. and Bryson. Optimal programming problems with inequality constraints *ii*: Solut ion by steepest ascent. *AIAA Journal*, 2(2):25 34, January 1964.
12. M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Intern ational Conference on Robotics and Automation*, page. 1419 1424. San Francisco, CA, April 7-10 1986.
13. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, (2):477 521, 1987.
14. P. Fiorini. *Robot Motion Planning among Moving Obstacles*. Ph D thesis, University of California, Los Angeles, January 1995.
15. P. Fiorini and Z. Shiller. Motion planning in dynamic envir onments using th e relative velocity paradigm. In *IEEE Internatio nal Conference of Automation and Robotics*, volume 1, p ages 560 566, May 1993.
16. .1.1). Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer G'r-i713/lie**. Addison-Wesley Publishing Com pany, Reading, MA, 1990.
17. K. Fujimura. On motion plan ming amidst transient obstacles. In *IEEE International Conference on Robotics and Automation*, pages 1488 1493, Nice, France, May 1992.
18. K. Fujimura. Motion planning amid transient obstacles. *International Journal of Robotics Research*, 13(5):395 407, October *1994*.
19. K. Fujimura and H. Samet. Time-mini mal paths am ong mov ing obstacles. In *IEEE International Conference on Robot ics and Automation*, pages 1110 1115, Scottsdale, AZ, May 1989.
20. K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE Int ernational Conference on Robotics and Automation*, pages 324 330, Cincinnati, O H, May 1990.
21. D.H. Jacobson, M.M. Lele, and J .L. Speyer. New necessar y conditions of optimality for control problems with state-variable inequality constraint s. *Journal of Mathematical A nalys is and Applicatio n s*, 35(2):255 284, *1971*.
22. D.W. Johnson and E. G. Gill ert. Minimum time robot planning in the presence of obstacles. In *IEEE Conference on Decision and Control*, pages *1748* 1753, Ft. Lauderdale, FL, December 1985.
23. K. Kant and S.W. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Jou rnal of Robotic Research*, 5(3):72 89, Fall 1986.
24. K. Kant and S. M'. Zucker. Planning collision free trajecto ries in time-varying environments: a two level hierarchy. In *IEEE International Conference on Robotics and Automation*, pages 1 6 4 4 1649. Raleigh, NC. 1988.
25. O. Khatib. A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Transaction on Robotics and Autom ation*, 3:43 53. 1987.
26. R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, (27):97 109, 1985.
27. B.H. Lee and C. S. G. Lee. Collision-free motion planning of two robots. *IEEE Transactions on System Man and Cybernetics*, SMC-17(1):21 32, January-February 1987.
28. G. Leitman. *An Introduction to Optimal Control*. McGraw-Hill Book Co., New York, NY, 1966.
29. T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Transaction on Computers*, C-32(2):108 120, February 1983.
30. T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free pat hs among polyhedral obstacles. *Communications of the ACM*, 22(10):560 570, October 1979.
31. E. B. Meier. *An Efficient Algorithm for Bang-Ban g Control Systems Applied to a Two-Link Manipulator*. PhD thesis, Stanford, December 1987.
32. N Nilsson. *Prin ciples of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.
33. C. Ó 'D únlaing. Motion plan ning with inertial constraints. Technical Report TR-230, NYU Robotics Laboratory, 1986.
34. J. Pearl. *Heuristics*. Addison Wesley Publishing, Co., Reading, MA, 1985.
35. J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *25th IEEE Symposium on the Foundation of Computer Science*, pages 144 153, 1985.
36. J.C. Sanborn and J.A. Hendler. A model of reaction for planning in dynamic environments. *In ternational Journal o r Artificial Intelligence in Engineering*, 3(2):95 101. April 1988.
37. Z. Shiller and Dubowsky. Time optimal paths and acceleration lines of robotic manipulators. Ia The *26th IEEE Conference on Decision and Control*, Los Angeles, CA, December 1987.
38. Z. Shiller and S. Dubowsky. Robot path planner with obstacles, actuators, gripper and payload constraints. *The Int ernational Journal of Robotics Research*, 8(6):3 18, December 1989.
39. K.G. Shin and N. T). McKay. Open loop minimum time control of mechanical manipulators. In *American Control Conference*, pages 1231 1236, June 1984.
40. J. Speyer. *Optimization and Con trol of Non lin ar Syst ems with Inflight Constraints*. PhD thesis, Harvard,

February 1968.

41. A. Weinreb and A. I?. Bryson. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, AG30((111 6), November 1985.